GIT AND GITHUB TUTORIAL

Git

- Git is a Version Control System.
- Version Control System is a tool that helps to track code changes.
- Git is Popular, Free, Open Source, Fast and Scalable.
- Git is used to Track the Code History.
- Git helps us to collaborate on projects.

Github

- GitHub is a Website that allows developers to store and manage their code using Git.
- https://github.com
- **Repositories** in GitHub act like a Folder in Systems.

Terminal Commands

- **cd** <folder name>: used to change folder inside the repository
- **mkdir** <folder name>: used to make a folder inside a repository
- **ls**: list down all files
- **ls -a**: shows all hidden file
- **cd ..:** Used to exit immediate directory

Create Repository on GitHub

- Create Account -> Go To Your Profile -> Select Repositories -> Select New
- Give Project Name -> Select as Public/Private -> Add a README file
- Select Create Repository

Setting up Git

- Download Visual Studio Code
- For Windows - Git Bash
- For Mac - Terminal

Configuring Git

- git config --global user. name "My Name"
- git config --global user.email "someone@gmail.com"
- git config --list

Git Command: Clone and Status

- Clone - Cloning a repository on our local machine
- **git clone** <some link>
- <Some link> -- GitHub -> repository --> code --> copy HTTPS
- Status - Display the state of the Code
- **git status**

4 types of Git Status

- **Untracked -** newly added files that git doesn't yet track
- **Modified -** changes made in the file
- **Staged -** file is ready to be committed
- **unmodified -** file is unchanged

Git Command: Add and Commit

- add - adds new or changed files in our working directory to the git staging area
- **git add** <file name>
- commit it is the record of the change
- **git commit -m** "some message”
- **First add(Staged)-Secondcommit-ThirdPush**

GitCommand: Push

- upload local repository content to the remote repository
- **Git push -u origin main**

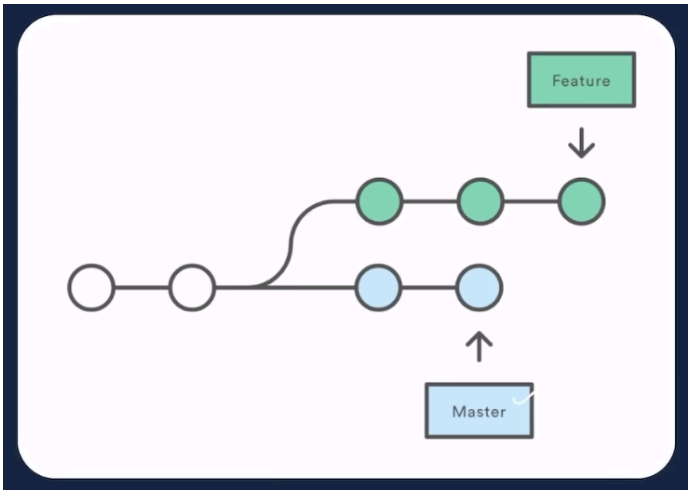
GitCommand: Init

- used to create a new git repo
- **git init -** adds new .git file inside the repository
- **gitremoteaddorigin**<link>
- **git remote -v** - to verify remote
- **git branch** - to check the branch
- **git branch -M main** - to rename branch
- **gitpushoriginmain**

Workflow

- **LocalGitWorkflow**
- **CreateGithubRepo->Clone->Changes->Add->Commit->Push**

Git Branch



Git Branch Commands

- **git branch** - to check the current branch
- **git branch -M** <new name> - to rename branch
- **git checkout** <branch name> - to navigate/change branch
- **git checkout -b** <new branch name> - to create new branch
- **git branch -d** <branch name> - to delete branch

Merging Git Branches

Way 1

- **git diff** <branch name> - to compare commits, branches, files and more
- **git merge** <breanch name> - to merge two branches

Way 2

Pull Request lets you tell others about changes that you've pushed to a branch in a repository on GitHub.

Pull Command

- **git pull origin main**
- Used to fetch and download content from a remote repo and immediately update the local repo to match that content

Resolving Merge Conflicts

An event that takes place when git is unable to resolve differences in code between two commits automatically.

Case1: Staged Changes

- **git reset** <file name>

Case2: Commit changes for one commit

- **git reset HEAD~1**

Case3: commit changes for many changes

- **git log** - shows git history
- Copy the previous commit hash code
- **git reset** <hashcode> - changes to the previous commit in git
- **git reset –hard** <hashcode> - changes to previous commit in local repo

Fork

A fork is anew repository that shares code and visibility settings with the original “upstream repository”. Fork is a rough copy.

- Search Project on GitHub
- Click on Fork
- Rename the repo and copy the master branch/ full project into our Github account
- Click on Create Fork
- Make Changes in the code (if necessary)
- Click on commit changes
- To Merge Our Commits With Original Project : Create Pull Request
- Click on New Pull Request

GITHUB CHEATSHEET

INSTALLATION & GUIS

With platform specific installers for Git, GitHub also provides the ease of staying up-to-date with the latest releases of the command line tool while providing a graphical user interface for day-to-day interaction, review, and repository synchronization.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

For Linux and Solaris platforms, the latest release is available on the official Git web site.

Git for All Platforms
http://git-scm.com

SETUP

Configuring user information used across all local repositories

`git config --global user.name “[firstname lastname]”`
set a name that is identifiable for credit when review version history
`git config --global user.email “[valid-email]”`
set an email address that will be associated with each history marker
`git config --global color.ui auto`
set automatic command line coloring for Git for easy reviewing

SETUP & INIT

Configuring user information, initializing and cloning repositories

`git init`
initialize an existing directory as a Git repository
`git clone [url]`
retrieve an entire repository from a hosted location via URL

STAGE & SNAPSHOT

Working with snapshots and the Git staging area

`git status`
show modified files in working directory, staged for your next commit
`git add [file]`
add a file as it looks now to your next commit (stage)
`git reset [file]`
unstage a file while retaining the changes in working directory
`git diff`
diff of what is changed but not staged
`git diff --staged`
diff of what is staged but not yet committed
`git commit -m “[descriptive message]”`
commit your staged content as a new commit snapshot

BRANCH & MERGE

Isolating work in branches, changing context, and integrating changes

`git branch`
list your branches. a * will appear next to the currently active branch
`git branch [branch-name]`
create a new branch at the current commit
`git checkout`
switch to another branch and check it out into your working directory
`git merge [branch]`
merge the specified branch’s history into the current one
`git log`
show all commits in the current branch’s history

INSPECT & COMPARE

Examining logs, diffs and object information

`git log`
show the commit history for the currently active branch
`git log branchB..branchA`
show the commits on branchA that are not on branchB
`git log --follow [file]`
show the commits that changed file, even across renames
`git diff branchB...branchA`
show the diff of what is in branchA that is not in branchB
`git show [SHA]`
show any object in Git in human-readable format

SHARE & UPDATE

Retrieving updates from another repository and updating local repos

`git remote add [alias] [url]`
add a git URL as an alias
`git fetch [alias]`
fetch down all the branches from that Git remote
`git merge [alias]/[branch]`
merge a remote branch into your current branch to bring it up to date
`git push [alias] [branch]`
Transmit local branch commits to the remote repository branch
`git pull`
fetch and merge any commits from the tracking remote branch

TRACKING PATH CHANGES

Versioning file removes and path changes

`git rm [file]`
delete the file from project and stage the removal for commit
`git mv [existing-path] [new-path]`
change an existing file path and stage the move
`git log --stat -M`
show all commit logs with indication of any paths that moved

REWRITE HISTORY

Rewriting branches, updating commits and clearing history

`git rebase [branch]`
apply any commits of current branch ahead of specified one
`git reset --hard [commit]`
clear staging area, rewrite working tree from specified commit

IGNORING PATTERNS

Preventing unintentional staging or commiting of files

`logs/`
`*.notes`
`pattern*/`

Save a file with desired patterns as .gitignore with either direct string matches or wildcard globs.
`git config --global core.excludesfile [file]`
system wide ignore pattern for all local repositories

TEMPORARY COMMITS

Temporarily store modified, tracked files in order to change branches

`git stash`
Save modified and staged changes
`git stash list`
list stack-order of stashed file changes
`git stash pop`
write working from top of stash stack
`git stash drop`
discard the changes from top of stash stack

Thank You !!!

Mail ID: 23f1001171@ds.study.iitm.ac.in